Depository Data Compression

Depository data compression is implemented on the PostgreSQL platform. It optimizes its structure and reduces its size several times. It is no longer possible to write to the compressed depository (not even with the arcsynchro utility). Data compression is available in D2000 version 21 and higher. Setting the parameter **TrezorCompress** = 1 enables compression of depository data after the depository is disconnected.

Even older depositories can be compressed "manually" using the TREZOR COMPRESS command. The opposite process is also supported - decompression using the TREZOR DECOMPRESS command.

The reading speed from compressed depositories is comparable to uncompressed ones. In some borderline cases (reading a specific value in time), compressed depositories may be slower. In other cases, reading compressed data may be faster (depending on the speed/usage ratio of the I/O subsystem and a processor). In addition, data compression allows you to store more data in memory (in the operating system cache and PostgreSQL).

It is possible to test working with compressed depositories. Setting the parameter **TrezorCompressKeep** = 1 causes tables with compressed data (CDATA, DATA0) to be created during compression, but the original table (DATA) is not deleted. After compression, the compressed data is then read. Using the TREZOR DECOMPRESS command, it is possible to return very quickly to the original structure of the depository, as it is not necessary to perform decompression, but simply delete the tables with compressed data.

Practical tests have shown that data compression in the depository is faster if the data is previously reorganized. Therefore, a TrezorCompressReorg parameter is preset to 1 and activates data reorganization before compression.

The list of depositories with the LIST_TREZOR command shows the *cps* flag for compressed depositories:

```
Received TELL command : LIST_TREZOR
List all
 List of existing trezors:
-------------------------------------------------------------------------
 ID | Data start(UTC)| Data end(UTC)  |     Status                      |
-------------------------------------------------------------------------
0134 31-05-2020 22:00 30-06-2020 22:00 being used for writing
0133 30-04-2020 22:00 31-05-2020 22:00 mounted read-only  cps
0132 31-03-2020 22:00 30-04-2020 22:00 mounted read-only  cps
0131 29-02-2020 23:00 31-03-2020 22:00 mounted read-only
0130 31-01-2020 23:00 29-02-2020 23:00 mounted read-only
```

PostgreSQL limitations

Compressed depositories on PostgreSQL are subject to the following limitation: the data size of one archive object (or one structured archive item) that is stored in one row of a compressed data table CDATA must be less than 1 GB. Otherwise, the compression ends with an error and the ID of the object that caused it is listed. Example:

```
[2021-04-05 20:36:52.371]E MES_TREZOR_20 ExecSqlCommand insert into cdata with avgc as (select '20090701
000000.000'::timestamp as cmin, '20090801 000000.000'::timestamp as cmax,'20090701 000000.000'::timestamp +
('20090801 000000.000'::timestamp - '20090701 000000.000'::timestamp)/2 as c), sdata as (SELECT * from data where
"ID"= 1309454 and "ROW"= 0 and "COL"= 0 order by "CAS") select "ID", "ROW", "COL", avgc.c, array_agg( ((extract
(epoch from "CAS"-avgc.c)*1000)::integer, "VALUE", "STATUS", "LIMIT_STATUS", "ARCHIV_STATUS", "FLAGS")::
d2trzitem) from sdata, avgc where "CAS" between avgc.cmin and avgc.cmax + '1 hour'::interval group by "ID",
"ROW", "COL", avgc.c
(54000)ERROR: array size exceeds the maximum allowed (1073741823);

[2021-04-05 20:36:52.381]E Error inserting cdata for 20 ID=$1309454
```

Configuration parameters of depository data compression for the PostgreSQL platform:

- **TrezorCompress** - parameter activates depository data compression. Data is compressed when the depository is disconnected.
  The parameter can be changed online, by the SET_OPTION TrezorCompress command.

- **TrezorCompressKeep** - parameter is used to test depository data compression. If set to 1, the original data table (DATA) will not be deleted but will be retained. However, the data will be read from tables with compressed data (CDATA, DATA0). The TREZOR DECOMPRESS command can be used to return very quickly to the original structure of the vault, as it is not necessary to perform decompression, but simply delete the tables with compressed data.
  The parameter can be changed online, by the SET_OPTION TrezorCompressKeep command.

  To **delete the compressed data and keep the original uncompressed data**, follow these steps:

  - Use the SET_OPTION TrezorCompressKeep OFF command to set the value of the **TrezorCompressKeep** parameter to 0.

  - Use the TREZOR DECOMPRESS command to decompress the depository data (decompression will take place quickly, as only tables with compressed data will be deleted - CDATA, DATA0).

  To **keep the compressed data and delete the original uncompressed data**, follow these steps:

  - Use the TREZOR DECOMPRESS command to decompress the depository data (decompression will take place quickly, as only the d epository will be marked as decompressed. Thanks to the value of the parameter **TrezorCompressKeep** = 1, the tables CDATA and DATA0 with compressed data will not be deleted).

- Use the SET_OPTION TrezorCompressKeep OFF command to set the value of the **TrezorCompressKeep** parameter to 0.

- Use the TREZOR COMPRESS command to compress the depository data (compression will take place quickly, as the existing data in the compressed data tables will be used). Compression will delete the original data table with uncompressed data (DATA).

- **TrezorCompressReorg** - reorganization of the depository data before compression. This parameter has a default value of 1 - in practice, it turns out that it is more efficient and faster to reorganize the data first (by an SQL command *CLUSTER DATA*) and then compress them. A value of 0 disables the reorganization of the depository data before compression. Disabling reorganization before compression is not recommended.
  The parameter can be changed online, by the SET_OPTION TrezorCompressReorg command.
- **TrezorCompressCmt** -the parameter specifies the number of archive objects after the compression of which a COMMIT will be executed. The default value is 10. In applications with intensively archived objects (many values of one archive object in the depository), this parameter can be reduced down to 1. Conversely, in applications with many archive objects that change infrequently, it is possible to increase the value of this parameter.
  The parameter can be changed online, by the SET_OPTION TrezorCompressCmt command.

- **TrezorCompressOrder** - the parameter refers to reading from depositories with compressed data. This parameter indicates whether time sorting is required when reading data from a compressed depository. Since the data is sorted during compression, it is possible to leave the parameter at 0 by default, which speeds up reading from the compressed vault (*ORDER BY* clause is omitted)
  The parameter can be changed online, by the SET_OPTION TrezorCompressOrder command.

The process of compressing old depositories

- If the old depositories have already been cleaned (e.g. the backup script contained the "cluster data" command), it is possible to turn off the cleaning with the following command before compression:
  SET_OPTION TREZOR_COMPRESS_REORG OFF
- The depositories to be compressed must be mounted for writing. Therefore, they must first be disconnected and then reconnected. For example for depositories 1-10:
  DISMOUNT_TREZOR 1 10
  MOUNT_TREZOR 1 10 WRITE
- It is now possible to compress the depositories with the command
  TREZOR COMPRESS <trezor_id>
  e.g. TREZOR_COMPRESS 1
  The depository with a size of about 20 GB is compressed on a standard server in about an hour or two. If it's much more, then the data is probably "cleaned" and cleaning needs to be turned on before compression (SET_OPTION TREZOR_COMPRESS_REORG ON).
  By default, the archive has the debug category DBG.ARCHIV.COMPRESS.TREZOR enabled, and debug logs are visible in the D2000 Sysconsole. They talk about the compression of concrete objects (or items of structured archives), in parentheses is the serial number / total number of objects). Once every 10 objects, a *commit* to the database follows (this can be parameterized with the command SET_OPTION TREZOR_COMPRESS_CMT <number>).
  Compress trezor 81 ID $11291(1048/ 39113)
  Compress trezor 81 ID $11291(1048/ 39113) done
  Compress trezor 81 ID $11293(1049/ 39113)
  Compress trezor 81 ID $11293(1049/ 39113) done
  Compress trezor 81 ID $11295(1050/ 39113)
  Compress trezor 81 ID $11295(1050/ 39113) done (commit)
  Compress trezor 81 ID $11297(1051/ 39113)
  Compress trezor 81 ID $11297(1051/ 39113) done
  Compress trezor 81 ID $11299(1052/ 39113)
  Compress trezor 81 ID $11299(1052/ 39113) done
- After compression, it is advisable to disconnect the depositories and connect them for reading. For example for depositories 1-10:
  DISMOUNT_TREZOR 1 10
  MOUNT_TREZOR 1 10
- With the command LIST_TREZOR, it is possible to check that all required depositories are compressed. The compressed depositories have a "cps" flag. For example, depository 140:
  0140 31-07-2019 22:00 31-08-2019 22:00 mounted read-only ***cps***
- If compression before cleaning was turned off, we recommend turning it on with the command:
  SET_OPTION TREZOR_COMPRESS_REORG OFF
- Compression of newly created depositories is turned on by command:
  SET_OPTION TREZOR_COMPRESS ON